(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2004-171167
(P2004-171167A)
(43)公開日　平成16年6月17日(2004.6.17)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 9/46	G06F 9/46	360B	5B042
G06F 11/34	G06F 11/34	M	5B045
G06F 15/177	G06F 15/177	672B	5B098

審査請求 有　請求項の数 8　OL　（全 16 頁）

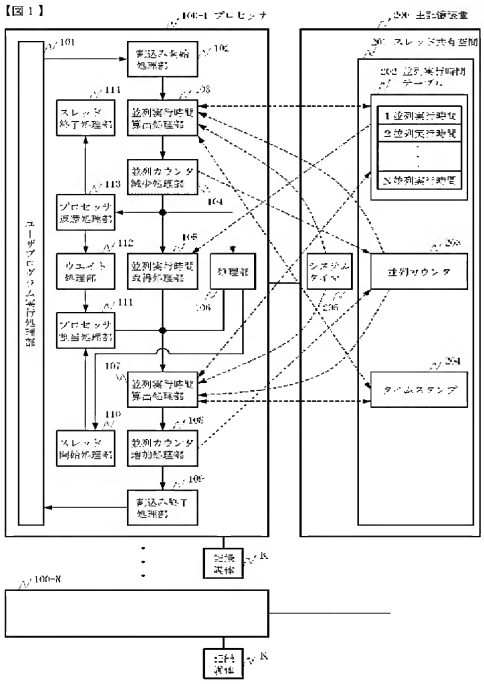
(21)出願番号　特願2002-334577(P2002-334577)
(22)出願日　平成14年11月19日(2002.11.19)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100088959
弁理士　境　廣巳
(72)発明者　田辺　純
東京都港区芝五丁目７番１号　日本電気株式会社内

Fターム(参考)　5B042 GA11 HH20 MA08
5B045 BB28 BB42 DD01 GG11 JJ02 JJ08
5B098 AA10 GA05 GC10 GD04

(54)【発明の名称】マルチプロセッサ計算機及びプログラム

(57)【要約】
【課題】マルチプロセッサ計算機上で複数の並列プログラムが動作している場合であっても、各並列プログラムの並列実行時間を取得できるようにする。
【解決手段】プロセッサ１００－ｎ（１≦ｎ≦Ｎ）は、自プロセッサでスレッドが生成されると、それがプロセス中の最初のスレッドであれば、上記憶装置２００上に上記プロセス固有の並列実行時間テーブル２０２を生成する。また、プロセッサ１００－ｎは、自プロセッサを上記プロセス内のスレッドに割り当てるとき、及び自プロセッサを上記プロセス内のスレッドから返還させるとき、システムタイマ２０５が示す現在時刻、タイムスタンプ２０４が示す並列実行時間テーブルの最新更新時刻、並列カウンタ２０３が示すスレッド数及び並列実行時間テーブル２０２の内容に基づいて並列実行時間テーブル２０２を更新する。スレッドは、並列実行時間取得システムコールにより並列実行時間テーブル２０２の内容を取得する。
【選択図】　図１



【特許請求の範囲】
【請求項１】
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、
前記各プロセッサが、
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有の並列実行時間テーブルであって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段と、
自プロセッサを前記プロセス内のスレッドに割り当てるとき、及び自プロセッサを前記プロセス内のスレッドから返還させるとき、現時点においてプロセッサが割り当てられている、前記プロセス内のスレッドの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段と、
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。
【請求項２】
請求項１記載のマルチプロセッサ計算機において、
前記並列実行時間テーブルが、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とするマルチプロセッサ計算機。
【請求項３】
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、
前記各プロセッサが、
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有のスレッド共有空間であって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられていた時間が登録される領域を有する並列実行時間テーブルと、前記プロセス内のスレッドの内の、現時点においてプロセッサが割り当てられているスレッドの数が設定される並列カウンタと、前記並列実行時間テーブルに対する最新の更新時刻が設定されるタイムスタンプとを含むスレッド共有空間を生成する手段と、
自プロセッサを前記プロセス内のスレッドに割り当てるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を増加させる手段と、
自プロセッサを前記プロセス内のスレッドから返還させるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を減少させる手段と、
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。
【請求項４】
請求項３記載のマルチプロセッサ計算機において、
前記スレッド共有空間が、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とするマルチプロセッサ計算機。
【請求項５】
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、
前記各プロセッサが、
自プロセッサで生成されたプロセスが、該プロセスが属するジョブ中の最初のプロセスで

ある場合、前記主記憶装置上に、前記ジョブに固有の並列実行時間テーブルであって、複数のプロセス数毎に、そのプロセス数以上のプロセスに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段と、
自プロセッサを前記ジョブ内のプロセスに割り当てるとき、及び自プロセッサを前記ジョブ内のプロセスから返還させるとき、現時点においてプロセッサが割り当てられている、前記ジョブ内のプロセスの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段と、
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。

【請求項６】
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、
前記各プロセッサが、
自プロセッサで生成されたプロセスが、該プロセスが属するジョブ中の最初のプロセスである場合、前記主記憶装置上に、前記ジョブに固有の共有空間であって、複数のプロセス数毎に、そのプロセス数以上のプロセスに同時にプロセッサが割り当てられていた時間が登録される領域を有する並列実行時間テーブルと、前記ジョブ内のプロセスの内の、現時点においてプロセッサが割り当てられているプロセスの数が設定される並列カウンタと、前記並列実行時間テーブルに対する最新の更新時刻が設定されるタイムスタンプとを含むプロセス共有空間を生成する手段と、
自プロセッサを前記ジョブ内のプロセスに割り当てるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を増加させる手段と、
自プロセッサを前記ジョブ内のプロセスから返還させるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を減少させる手段と、
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とするマルチプロセッサ計算機。

【請求項７】
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機の構成要素であるプロセッサを、
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有の並列実行時間テーブルであって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段、
自プロセッサを前記プロセス内のスレッドに割り当てるとき、及び自プロセッサを前記プロセス内のスレッドから返還させるとき、現時点においてプロセッサが割り当てられている、前記プロセス内のスレッドの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段、
前記並列実行時間テーブルの内容を取得する手段として機能させるためのプログラム。

【請求項８】
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機の構成要素であるプロセッサを、
自プロセッサで生成されたプロセスが、該プロセスが属するジョブ中の最初のプロセスである場合、前記主記憶装置上に、前記ジョブに固有の並列実行時間テーブルであって、複数のプロセス数毎に、そのプロセス数以上のプロセスに同時にプロセッサが割り当てられ

た時間を登録する領域を有する並列実行時間テーブルを生成する手段、
自プロセッサを前記ジョブ内のプロセスに割り当てるとき、及び自プロセッサを前記ジョブ内のプロセスから返還させるとき、現時点においてプロセッサが割り当てられている、前記ジョブ内のプロセスの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段、
前記並列実行時間テーブルの内容を取得する手段として機能させるためのプログラム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、複数のプロセッサが主記憶装置を共有するマルチプロセッサ計算機に関し、特に、並列プログラムが如何に効率良く動作したかを表す指標である並列実行時間を算出する技術に関する。
【０００２】
【従来の技術】
複数のプロセッサから構成されるマルチプロセッサ計算機の性能評価を行う従来の技術として、各プロセッサ毎にプロセッサが稼働状態にあるときの開始時刻及び終了時刻と、プロセッサが非稼働状態のときの時刻とを記録し、制御部が、上記各時刻に基づいて任意の時刻におけるプロセッサの稼働台数や、稼働率を求める技術が知られている（例えば、特許文献１参照）。
【０００３】
【特許文献１】
特開平５－１８９３９５号公報
【０００４】
【発明が解決しようとする課題】
上述した従来の技術によれば、何台のプロセッサが同時に稼働しているかを知ることはできるが、マルチプロセッサ計算機を使用して並列プログラムを実行するユーザにとって重要なことは、プロセッサが何台同時に稼働しているかではなく、並列プログラムが効率的に実行されるか否かである。マルチプロセッサ計算機で実行される並列プログラムが１つの場合には、上述した従来の技術によっても並列プログラムが効率的に実行されているか否かを判定することができる。つまり、多くのプロセッサが同時に稼働している時間が長いほど、並列プログラムが効率的に実行されていると判定することができる。
【０００５】
しかし、マルチプロセッサ計算機で実行される並列プログラムが複数の場合は、プロセッサの稼働台数が分かったとしても、稼働台数に基づいて並列プログラムが効率的に実行されているか否かを判定することはできない。例えば、並列プログラムαの２つのスレッドと、並列プログラムβの１つのスレッドとがマルチプロセッサ計算機で実行されている場合、上述した従来の技術では、プロセッサの稼働台数が３台であることが分かるだけであり、並列プログラムα，β毎に、同時に実行されているスレッド数が分からないため、各並列プログラムが効率的に実行されているか否かを判定することはできない。
【０００６】
そこで、本発明の目的は、複数の並列プログラムが同時に実行されている場合でも、各並列プログラムが効率的に実行されているか否かを判定できるようにすることにある。
【０００７】
【課題を解決するための手段】
本発明にかかる第１のマルチプロセッサ計算機は、上記目的を達成するため、複数のプロセッサと、該複数のプロセッサによって共有される上記憶装置とを備えたマルチプロセッサ計算機において、
前記各プロセッサが、
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッド

である場合、前記主記憶装置上に、前記プロセスに固有の並列実行時間テーブルであって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを生成する手段と、
自プロセッサを前記プロセス内のスレッドに割り当てるとき、及び自プロセッサを前記プロセス内のスレッドから返還させるとき、現時点においてプロセッサが割り当てられている、前記プロセス内のスレッドの数と、現時点の時刻と、現時点における前記並列実行時間テーブルの内容と、前記並列実行時間テーブルに対する最新の更新時刻とに基づいて、前記並列実行時間テーブルを更新する手段と、
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とする。

【０００８】

また、本発明にかかる第２のマルチプロセッサ計算機は、
第１のマルチプロセッサ計算機において、
前記並列実行時間テーブルが、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とする。

【０００９】

より具体的には、本発明にかかる第３のマルチプロセッサ計算機は、
複数のプロセッサと、該複数のプロセッサによって共有される主記憶装置とを備えたマルチプロセッサ計算機において、
前記各プロセッサが、
自プロセッサで生成されたスレッドが、該スレッドが属するプロセス中の最初のスレッドである場合、前記主記憶装置上に、前記プロセスに固有のスレッド共有空間であって、複数のスレッド数毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられていた時間が登録される領域を有する並列実行時間テーブルと、前記プロセス内のスレッドの内の、現時点においてプロセッサが割り当てられているスレッドの数が設定される並列カウンタと、前記並列実行時間テーブルに対する最新の更新時刻が設定されるタイムスタンプとを含むスレッド共有空間を生成する手段と、
自プロセッサを前記プロセス内のスレッドに割り当てるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を増加させる手段と、
自プロセッサを前記プロセス内のスレッドから返還させるとき、現在時刻と、前記並列実行時間テーブルの内容と、前記並列カウンタの内容と、前記タイムスタンプの内容とに基づいて、前記並列実行時間テーブルの内容を更新すると共に、前記タイムスタンプに現在時刻を設定し、更に、前記並列カウンタの値を減少させる手段と、
前記並列実行時間テーブルの内容を取得する手段とを備えたことを特徴とする。

【００１０】

本発明にかかる第４のマルチプロセッサ計算機は、
第３のマルチプロセッサ計算機において、
前記スレッド共有空間が、前記主記憶装置以外の、前記複数のプロセッサによって共有される記憶装置上に生成されることを特徴とする。

【００１１】

【作用】

或るプロセッサでスレッドが生成されると、そのスレッドがプロセス中の最初のスレッドである場合、主記憶装置上に上記プロセス固有の並列実行時間テーブルが作成される。その後、何れかのプロセッサが、上記プロセス内のスレッドにプロセッサを割り当てたり、上記プロセス内のスレッドからプロセッサを返還させるとき、現時点の時刻と、上記並列実行時間テーブルに対する最新の更新時刻と、現時点においてプロセッサが割り当てられている上記プロセスに属するスレッドの数と、並列実行時間テーブルの内容とに基づいて並列実行時間テーブルを更新する。このように、プロセス固有の並列実行時間テーブルを用いて並列実行時間を管理するようにしているので、マルチプロセッサ計算機で複数の並

列プログラムが同時に実行されている場合であっても、各並列プログラムが効率的に動作しているか否かを判定することが可能になる。

【００１２】

【発明の実施の形態】

次に本発明の実施の形態について図面を参照して詳細に説明する。

【００１３】

【実施例の構成】

図１は本発明にかかるマルチプロセッサ計算機の実施例のブロック図であり、複数のプロセッサ（ＣＰＵ，演算器）１００－１～１００－Ｎと、各プロセッサ１００－１～１００－Ｎによって共有される主記憶装置２００とから構成されている。

【００１４】

先ず、以下の説明で使用する用語の定義について述べておく。

【００１５】

プロセスとは、ＵＮＩＸ（登録商標）システムにおいて一般的にプロセスと呼ばれる概念に相当する。すなわち、プロセスとは、固有の記憶領域を有する最小の実行単位であり、時分割処理によりスケジューリングされるものである。従って、マルチプロセッサ計算機においては、複数のプロセッサ上で同時に異なるプロセスを実行することができ、シングルプロセッサシステムにおいても複数のプロセスをインタラクティブに実行することができる。

【００１６】

また、スレッドとは、ＵＮＩＸ（登録商標）システムにおいて一般的にスレッドと呼ばれる概念に相当する。すなわち、スレッドは単一のプロセスから生成された複数の実行単位であり、プロセスと同様に時分割処理によりスケジューリングされる。スレッドは固有の記憶領域を持たない最小の実行単位であり、同一プロセスに属する各スレッドはそれらが属している同一プロセスに固有の記憶領域を共有している。

【００１７】

また、時分割処理とは、非同期的あるいは同期的に発生する割り込みを契機として、実行待ち状態のプロセスのうち実行優先度の高いものからプロセッサが割り当てられ、プロセッサ上で実行される間に実行優先度が低下し、優先度が低いものからプロセッサを返還して実行待ち状態に移行するスケジューリング方式のことである。一般的にはマルチプロセッサ計算機における並列プログラムの最小実行単位はスレッドにより実現される事が多いが、プロセスによって実現される場合もある。スレッドにより実現される場合は、並列プログラムは複数のスレッドにより構成されたプロセスに相当し、このプロセスに固有の空間でかつこのプロセスを構成する複数のスレッドが互いに共有するスレッド共有空間を利用して動作する。プロセスにより実現される場合は、並列プログラムは複数のプロセスにより構成されたジョブに相当し、このジョブに固有の空間でかつこのジョブを構成する複数のプロセスが互いに共有するプロセス共有空間を利用して動作する。以下では並列プログラムの最小の実行単位がスレッドである事を想定した記述となっているが、最小の実行単位がプロセスであるようなシステムの場合でも、プロセスをスレッド、ジョブをプロセスと読み替えることで同様の効果を得ることができる。

【００１８】

図１において、主記憶装置２００は、各プロセッサ１００－１～１００－Ｎから等価に共有されたシステムに唯一の主記憶装置である。主記憶装置２００は、スレッド共有空間２０１と、システムタイマ２０５とを含んでいる。

【００１９】

スレッド共有空間２０１は、任意のある時点においてシステムに存在する複数のプロセスの内の任意の一つが占有してアクセス可能な記憶領域であり、なおかつそのプロセスに属する各スレッドが共有している記憶領域である。このようなスレッド共有空間２０１には、並列実行時間テーブル２０２と、並列カウンタ２０３と、タイムスタンプ２０４とが設けられている。

【００２０】
並列実行時間テーブル２０２は、対応するプロセスを構成しているスレッドに関する１並列実行時間からＮ並列実行時間までの配列（要素数Ｎ）が格納される領域を表している。ここで、ｎ並列実行時間（ｎ＝１，２，…，Ｎ）とはｎ個以上のスレッドが同時に動作していた時間（ｎ個以上のスレッドに同時にプロセッサが割り当てられていた時間）を表すものとする。
【００２１】
並列カウンタ２０３は、対応するプロセスを構成するスレッドの内、同時に異なるプロセッサで動作しているスレッドの数（０～Ｎ）を表している。
【００２２】
タイムスタンプ２０４には、並列実行時間テーブル２０２に対する最新の更新時刻が格納される。
【００２３】
システムタイマ２０５には、システム稼働中にマシンクロック単位で単調増加し続けるシステムに唯一のタイマ値が格納される。
【００２４】
また、図１においてプロセッサ１００－１は、ユーザプログラム実行処理部１０１と、割込み開始処理部１０２と、並列実行時間算出処理部１０３と、並列カウンタ減少処理部１０４と、並列実行時間取得処理部１０５と、処理部１０６と、並列実行時間算出処理部１０７と、並列カウンタ増加処理部１０８と、割込み終了処理部１０９と、スレッド開始処理部１１０と、プロセッサ割当処理部１１１と、ウェイト処理部１１２と、プロセッサ返還処理部１１３と、スレッド終了処理部１１４と、記録媒体Ｋとを備えている。なお、他のプロセッサもプロセッサ１００－１と同様の構成を有している。
【００２５】
ユーザプログラム実行処理部１０１は、スレッドがユーザモードでプログラムのコードを実行する手段である。
【００２６】
割込み開始処理部１０２は、システムコールやＩ／Ｏ処理、例外処理などの非同期ないし同期的な割込みの発生に伴い、プロセッサコンテキストのセーブや割込み原因に対応する割込みハンドラを呼び出す。
【００２７】
並列実行時間算出部１０３は、割込み開始処理の直後に呼び出され、システムコールの発行元のスレッドが属するプロセスに固有のスレッド共有空間２０１内の並列カウンタ２０３の値を参照し、その値が、「１」以上であった場合は、並列実行時間テーブル２０２内の第１番目のエントリから並列カウンタ２０３の値に対応するエントリまでの各値に、システムタイマ２０５とタイムスタンプ２０４との差分を加算し、その後、タイムスタンプ２０４にシステムタイマ２０５の値を設定する。また、並列カウンタ２０３の値が「０」であった場合は、タイムスタンプ２０４にシステムタイマ２０５の値を設定する処理のみを行い、並列実行時間テーブル２０２に対する更新処理は行わない。
【００２８】
並列カウンタ減少処理部１０４は、並列カウンタ２０３の値を１減少させる機能を有する。
【００２９】
並列実行時間取得処理部１０５は、並列実行時間取得システムコールにより呼び出され、発行元のスレッドが属するプロセスに対応するスレッド共有空間２０１の並列実行時間テーブル２０２の各エントリの内容を取得し、スレッドに返却する機能を有する。
【００３０】
並列実行時間算出処理部１０７は、並列実行時間算出処理部１０３と同様の機能を有する。
【００３１】

並列カウンタ増加処理部１０８は、並列カウンタ２０３の値を１増加させる機能を有する。

【００３２】
割込み終了処理部１０９は、プロセッサコンテキストのリストアや割込み発生ポイントへの復帰を行う。

【００３３】
スレッド開始処理部１１０は、新たに生成されたスレッドが、そのスレッドの属しているプロセス内の最初のスレッドであった場合、プロセスに固有の空間で且つそのプロセスに属している全てのスレッドが共有するスレッド共有空間２０１を確保した上で、並列実行時間テーブル２０２、並列カウンタ２０３及びタイムスタンプ２０４の値を「０」に初期化する。

【００３４】
プロセッサ割当処理部１１１は、他の実行優先度の低いスレッドからプロセッサを奪い取り、仮想空間の切り替えなどのスイッチ処理を実行してスレッドの実行を開始させる。

【００３５】
ウェイト処理部１１２は、スリープシステムコールやタイムスライス切れが発生した場合、スレッドを実行待ちキューやスリープキューなどを用いて停止させる。

【００３６】
プロセッサ返還処理部１１３は、システムコールやＩ／Ｏ待ちによるスリープ呼び出しや、終了処理、タイムスライス切れ割込みで自スレッドよりも実行優先度の高いスレッドがある時などに呼び出され、他の実行優先度の高いスレッドにプロセッサを譲るために仮想空間の切り替えなどのスイッチ処理を実行する。

【００３７】
スレッド終了処理部１１４は、終了したスレッドがプロセス内の最後のスレッドである場合は、上記プロセス固有のスレッド共有空間２０１を解放する機能を有する。

【００３８】
なお、並列実行時間算出処理部１０３，並列カウンタ減少処理部１０４によって連続的に行われる処理と、並列実行時間算出処理部１０７，並列カウンタ増加処理部１０８によって連続的に行われる処理と、並列実行時間取得処理部１０５によって行われる処理とはそれぞれ互いに、同一プロセス内の異なるスレッドにおいて同時に実行されないよう、排他制御によりシリアライズされている。

【００３９】
処理部１０６は、プロセッサ返還処理、並列実行時間取得処理以外の、システムコールに応じた処理、Ｉ／Ｏ処理、例外処理などの一般的な割込み処理を行う。

【００４０】
記録媒体Ｋは、ディスク、半導体メモリ、その他の記録媒体である。この記録媒体Ｋに記録されているプログラムは、プロセッサ１００－１によって読み取られ、その動作を制御することで、プロセッサ１００－１上に、ユーザプログラム実行処理部１０１、割込み開始処理部１０２、並列実行時間算出処理部１０３、並列カウンタ減少処理部１０４、並列実行時間取得処理部１０５、処理部１０６、並列実行時間算出処理部１０７、並列カウンタ増加処理部１０８、割込み終了処理部１０９、スレッド開始処理部１１０、プロセッサ割当処理部１１１、ウェイト処理部１１２、プロセッサ返還処理部１１３、スレッド終了処理部１１４を実現する。

【００４１】
【実施例の動作】
次に各図を参照して本実施例の動作について詳細に説明する。

【００４２】
先ず、プロセッサ１００－１においてスレッドＸが生成されると、プロセッサ１００－１内のスレッド開始処理部１１０は、スレッドＸを実行待ちキュー（図示せず）につなぎ、更に、図２の流れ図に示すように、スレッドＸが、それが属するプロセスＺ内の最初のス

レッドであるか否かを判断し（ステップＳ１）、最初のスレッドであった場合（ステップＳ１がＹｅｓ）は、上記プロセスＺに固有のスレッド共有空間２０１を生成し、更に、並列実行時間テーブル２０２の各エントリ、並列カウンタ２０３及びタイムスタンプ２０４の値を「０」に初期設定する（ステップＳ２）。これに対して、最初のスレッドでなかった場合（ステップＳ１がＮｏ）は、ステップＳ２をスキップする。

【００４３】

その後、或るプロセッサ（例えば、プロセッサ１００－１）内のプロセッサ割当処理部１１１によって、実行待ちキューにつながれているスレッドＸにプロセッサ１００－１が割り当てられると、並列実行時間算出処理部１０７が、並列実行時間の算出処理を行う（ステップＳ１３）。

【００４４】

このステップＳ１３の処理を詳しく説明すると、次のようになる。先ず、スレッドＸが属するプロセスＺに割り当てられているスレッド共有空間２０１中の並列カウンタ２０３の値Ｃを参照し、その値Ｃが「１」以上であるか否かを調べる。

【００４５】

そして、「１」以上であった場合は、システムタイマ２０５の値ＳＴＭとタイムスタンプ２０４の値ＴＳとの差分（ＳＴＭ－ＴＳ）を求める。その後、並列実行時間テーブル２０２の各エントリの内の、第１番目のエントリから並列カウンタ２０３の値に対応するエントリまでを対象にして、そこに設定されている値に上記差分（ＳＴＭ－ＴＳ）を加算する。そして、最後に、タイムスタンプ２０４の値ＴＳにシステムタイマ２０５の値ＳＴＭを設定する。

【００４６】

これに対して、並列カウンタ２０３の値Ｃが「１」未満であった場合は、タイムスタンプ２０４の値ＴＳにシステムタイマ２０５の値ＳＴＭを設定する処理のみを行い、並列実行時間テーブル２０２に対する更新処理は行わない。以上がステップＳ１３で行う処理の詳細である。

【００４７】

並列実行時間算出処理部１０７の処理が終了すると、並列カウンタ増加処理部１０８が、スレッドＸが属しているプロセスＺに割り当てられているスレッド共有空間２０１内の並列カウンタ２０３の値Ｃを増加させる（ステップＳ１４）。なお、本実施例では、並列カウンタ２０３の値Ｃを「１」増加させるものとする。

【００４８】

その後、割込み終了処理部１０９による割込み終了処理が行われ、上記スレッドＸは、ユーザモードに移行し、ユーザプログラム実行処理部１０１でユーザプログラムを実行する（ステップＳ３）。このステップＳ３では、並列プログラムがプログラムの記述に従って実行され、割込みが発生しない限り（ステップＳ４がＹｅｓとならない限り）、ユーザプログラムが実行され続ける。

【００４９】

スレッドＸがシステムコールやＩ／Ｏを発行したり、タイムスライス切れや例外処理により割り込まれた場合（ステップＳ４がＹｅｓ）は、割込み開始処理部１０２に処理が移行し、以下の処理が行われる。

【００５０】

先ず、並列実行時間算出処理部１０３がスレッドＸが属しているプロセスＺのスレッド共有空間２０１を対象にして、ステップＳ１３において並列実行時間算出処理部１０７が行った処理と同様の処理を行う（ステップＳ７）。次いで、並列カウンタ減少処理部１０４が並列カウンタ２０３の値Ｃを減少させる（ステップＳ８）。本実施例では、並列カウンタ２０３の値Ｃを「１」減少させるものとする。

【００５１】

その後、割込み要因に対応する割込み処理が実行される。もし、割込み要因が並列実行時間取得システムコールである場合（ステップＳ９がＹｅｓ）は、並列実行時間取得処理部

１０５において、並列実行時間の取得処理が行われる（ステップＳ１１）。このステップＳ１１の並列実行時間の取得処理では、並列実行時間テーブル２０２の内容を全て取得し、システムコールの発行元のスレッドに返却する処理が行われる。その後、並列実行時間算出処理部１０７において、スレッドＸが属するプロセスＺのスレッド共有空間２０１を処理対象にして前述した並列実行時間の算出処理が行われ（ステップＳ１３）、更に、並列カウンタ増加処理部１０８において前述した並列カウンタ２０３の増加処理が行われる（ステップＳ１４）。その後、割込み終了処理部１０９において割込み終了処理が実行され、上記スレッドＸは、ユーザモードに移行し、ユーザプログラム実行処理部１０１でユーザプログラムを実行する（ステップＳ３）。

【００５２】

また、割込み要因が並列実行時間取得システムコールでなかった場合（ステップＳ９がＮｏ）は、割込み要因が終了システムコールであるか否かを判定する（ステップＳ１０）。

【００５３】

そして、終了システムコールでなかった場合（ステップＳ１０がＮｏ）は、処理部１０６において割込み要因に応じた割込み処理が行われる（ステップＳ１２）。このステップＳ１２で行われる処理としては、例えば、新たなスレッドＹの生成処理などがある。もし、ステップＳ１２で新たなスレッドＹが生成されたとすると、スレッド開始処理部１１０は、スレッドＹを実行待ちキューにつなぐと共に、上記スレッドＹを処理対象にして前述したステップＳ１の処理を行う。この場合、スレッドＹはプロセスＺ中の最初のスレッドでないので（ステップＳ１がＮｏ）、ステップＳ２はスキップされる。その後、プロセッサ１００－１～１００－Ｎの内の、或る１台のプロセッサ１００－ｎ（１≦ｎ≦Ｎ）内のプロセッサ割当処理部１１１によってスレッドＹにプロセッサ１００－ｎが割り当てられると、並列実行時間算出処理部１０７、並列カウンタ増加処理部１０８、割込み終了処理部１０９による処理が順次行われる。

【００５４】

これに対して、割込み要因が終了システムコールであった場合（ステップＳ１０がＹｅｓ）は、プロセッサ返還処理部１１３によるプロセッサ返還処理が行われた後、スレッド終了処理部１１４によるスレッド終了処理が行われる。このスレッド終了処理においては、先ず、終了システムコールの発行元のスレッド（例えば、スレッドＸ）が、同一プロセスＺ内の最後のスレッドであるか否かを判断する（ステップＳ５）。そして、最後のスレッドである場合（ステップＳ５がＹｅｓ）は、スレッドＸが属しているプロセスＺに割り当てられているスレッド共有空間２０１を解放する（ステップＳ６）。これに対して、最後のスレッドでない場合（ステップＳ５がＮｏ）は、ステップＳ６をスキップする。その後、スレッドＸは終了する。

【００５５】

以上の処理により、割込みの発生毎に並列実行時間テーブル２０２に１並列実行時間からＮ並列実行時間までの各値が適時採取されことになる。なお、並列実行時間をＴｎ、並列カウンタ２０３の値をＣ、タイムスタンプ２０４の値をＴＳ、システムタイマ２０５の値をＳＴＭと表記すると、ステップＳ７、Ｓ１３の並列実行時間の算出処理は、Ｔｎ＝Ｔｎ＋（ＳＴＭ－ＴＳ）（但し１≦ｎ≦Ｃの各値について）、およびＴＳ＝ＳＴＭ　と表記できる。また、ステップＳ８の並列カウンタ減算処理はＣ＝Ｃ－１と表記でき、ステップＳ１４の並列カウンタ増加処理は　Ｃ＝Ｃ＋１と表記できる。

【００５６】

次に、具体例を用いて並列実行時間が算出される様子を説明する。

【００５７】

図３は、３つのスレッドＡ、Ｂ、Ｃにより構成される並列プログラムが動作した際の、並列実行時間テーブル２０２の各エントリの値Ｔｎ（ｎ＝１，２，…，Ｎ）、並列カウンタ２０３の値Ｃ及びタイムスタンプ２０４の値ＴＳの遷移を表した図である。なお、以下では説明を容易にするために、割り込みは時刻Ｐ１，Ｐ２，…，Ｐ８の８箇所においてのみ発生したものとし、なおかつ割り込みが発生してから終了するまでに要する時間は十分に

短く０に等しかったと仮定する。すなわち、割込み開始処理部１０２で割込み開始処理が行われてから割込み終了処理部１０９で割込み終了処理が行われるまでの間にシステムタイマ２０５の値は変動しなかったと仮定する。なお、現実のシステムおいては、割り込みは任意のスレッドの任意の時点で発生し、割り込みが発生してから終了するまでに要する時間は０以上の可変値となるが、現実のシステムのどのような場合も図３で例示した処理の組み合わせにすぎず、並列実行時間が正しく採取される。

【００５８】

時刻Ｐ１（ＳＴＭ＝１０）において、プロセッサ１００－１でスレッドＡが生成されると、プロセッサ１００－１内のスレッド開始処理部１１０は、スレッドＡを実行待ちキューにつなぐと共に、スレッドＡが、それが属するプロセスＤ内の最初のスレッドであるか否かを判断する（図２、ステップＳ１）。この場合、スレッドＡは最初のスレッドであるので（ステップＳ１がＹｅｓ）、スレッド開始処理部１１０は、スレッドＡが属するプロセスＤ固有のスレッド共有空間２０１を生成し、更に、並列実行時間テーブル２０２の各エントリの値Ｔ１～ＴＮ、並列カウンタ２０３の値Ｃ及びタイムスタンプ２０４の値ＴＳを全て「０」に初期化する（ステップＳ２）。

【００５９】

その後、或るプロセッサ（例えば、プロセッサ１００－１）内のプロセッサ割当処理部１１１によって、実行待ちキューにつながれているスレッドＡにプロセッサ１００－１が割り当てられると、並列実行時間算出処理部１０７が、ステップＳ１３の並列実行時間算出処理を行う。この場合、並列カウンタ２０３の値Ｃは「０」であるので、並列実行時間算出処理部１０７は、並列実行時間テーブル２０２に対する更新処理は行わず、タイムスタンプ２０４の値ＴＳにシステムタイマ２０５の値ＳＴＭの値を設定する処理だけを行う。従って、ＴＳ＝１０となる。

【００６０】

その後、並列カウンタ増加処理部１０８が、並列カウンタ２０３の値Ｃを＋１し、Ｃ＝１とする（ステップＳ１４）。その後、割込み終了処理部１０９による割込み終了処理が行われ、スレッドＡがユーザプログラム実行処理部１０１でユーザプログラムの実行を開始する（ステップＳ３）。

【００６１】

次に、時刻Ｐ２（ＳＴＭ＝２５）において、スレッドＡが新たなスレッドＢを生成するためにスレッド生成システムコールを発行する（ステップＳ４がＹｅｓ）。これにより、割込み開始処理部１０２に処理が移行し、並列実行時間算出処理部１０３において並列実行時間算出処理が行われる（ステップＳ７）。このとき、並列カウンタ２０３の値Ｃは「１」であるので、並列実行時間テーブル２０２の第１番目のエントリの値Ｔ１に、システムタイマ２０５の値ＳＴＭとタイムスタンプ２０４の値ＴＳとの差分（ＳＴＭ－ＴＳ）を加算し、更に、システムタイマ２０５の値ＳＴＭをタイムスタンプ２０４の値ＴＳに設定する。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝０＋（２５－１０）＝１５、ＴＳ＝ＳＴＭ＝２５となる。その後、並列カウンタ減少処理部１０４が、並列カウンタ２０３の値Ｃを１減少させ、Ｃ＝０とする（ステップＳ８）。

【００６２】

その後、プロセッサ１００－１内の処理部１０６においてスレッドＢが生成される（ステップＳ１２）。

【００６３】

スレッドＢが生成されると、プロセッサ１００－１内のスレッド開始処理部１１０は、スレッドＢを実行待ちキューにつなぐと共に、スレッドＢがプロセスＤの最初のスレッドであるか否かを判断する（ステップＳ１）。この場合、スレッドＢは最初のスレッドでないので、ステップＳ２の処理はスキップされる。

【００６４】

また、プロセッサ１００－１内の並列実行時間算出処理部１０７において、並列実行時間算出処理が行われる（ステップＳ１３）。このとき、並列カウンタ２０３の値Ｃは「０」

であるので、並列実行時間テーブル２０２は更新されず、タイムスタンプ２０４の値ＴＳにシステムタイマ２０５の値ＳＴＭが設定される。すなわち、ＴＳ＝２５となる（ＳＴＭは変動しなかったと仮定）。その後、並列カウンタ増加処理部１０８において、並列カウンタ増加処理が行われ、並列カウンタ２０３の値Ｃが「１」にされる（ステップＳ１４）。その後、割込み終了処理部１０９による割込み終了処理が行われ、スレッドＡはユーザプログラムの処理に復帰する。

【００６５】

一方、実行待ちキューにつながれたスレッドＢ（新たに生成されたスレッド）に、或るプロセッサ（例えば、プロセッサ１００－２）内のプロセッサ割当処理部１１１によりプロセッサ１００－２が割り当てられると、プロセッサ１００－２内の並列実行時間算出処理部１０７が並列実行時間の算出処理を行う（ステップＳ１３）。この時、スレッドＢが属するプロセスＤに割り当てられているスレッド共有空間２０１内の並列カウンタ２０３の値Ｃは「１」であるので、並列実行時間テーブル２０２の第１番目のエントリの値Ｔ１をＴ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝１５＋（２５－２５）＝１５とし、タイムスタンプ２０４の値ＴＳを「２５」とする。その後、並列カウンタ増加処理部１０８において、並列カウンタ２０３に対する増加処理が行われ、並列カウンタ２０３の値Ｃが「２」にされる。

【００６６】

その後、割込み終了処理部１０９による割込み終了処理が行われ、スレッドＢは、ユーザプログラムの実行を開始する。ここで、スレッドＡが割り当てられているプロセッサ１００－１内の並列実行時間算出処理部１０７、並列カウンタ増加処理部１０８よりも、スレッドＢが割り当てられるプロセッサ１００－２内の並列実行時間算出処理部１０７、割込み終了処理部１０９が先の処理を行ったとしても最終的な結果は同じである。

【００６７】

次に、時刻Ｐ３（ＳＴＭ＝３５）において、プロセッサ１００－２で実行されているスレッドＢがスリープシステムコールを発行すると、プロセッサ１００－２内の割込み開始処理部１０２に制御が移り、並列実行時間算出処理部１０３が並列実行時間の算出処理を行う（ステップＳ７）。この時、並列カウンタ２０３の値Ｃは「２」であるので、並列実行時間テーブル２０２の第１番目、第２番目の値Ｔ１、Ｔ２が更新される。すなわち、ＳＴＭ－ＴＳ＝３５－２５＝１０が、値Ｔ１、Ｔ２に加算され、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝１５＋１０＝２５、Ｔ２＝Ｔ２＋（ＳＴＭ－ＴＳ）＝０＋１０＝１０となる。その後、並列実行時間算出処理部１０３において並列カウンタ２０３の減少処理が行われ、並列カウンタ２０３の値Ｃが「１」にされる（ステップＳ８）。その後、プロセッサ返還処理部１１３がプロセッサの返還処理を行い、更に、ウェイト処理部１１２がスレッドＢをスリープキューにつなぎ、停止させる。

【００６８】

次に、時刻Ｐ４（ＳＴＭ＝４０）において、例えば、プロセッサ１００－２内のプロセッサ割当処理部１１１が、停止していたスレッドＢにプロセッサ１００－２を割り当てると、プロセッサ１００－２内の並列実行時間算出処理部１０７が、並列実行時間の算出処理を行う（ステップＳ１３）。この時、並列カウンタ２０３の値Ｃは「１」となっているので、並列実行時間算出処理部１０７は、並列実行時間テーブル２０２の第１番目のエントリの値Ｔ１を更新すると共に、タイムスタンプ２０４の値ＴＳを更新する。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝２５＋（４０－３５）＝３０、ＴＳ＝ＳＴＭ＝４０とする。その後、並列カウンタ増加処理部１０８が、並列カウンタ２０３の値Ｃを＋１して「２」にする。その後、割込み終了処理部１０９により割込み終了処理が行われ、スレッドＢがユーザプログラムの実行を再開する。

【００６９】

次に、時刻Ｐ５（ＳＴＭ＝５０）において、プロセッサ１００－２で動作しているスレッドＢが、スレッドＣを生成するためにスレッド生成システムコールを発行すると、プロセッサ１００－２内の割込み開始処理部１０２に制御が移る。これにより、並列実行時間算出処理部１０３が並列実行時間の算出処理を行う（ステップＳ７）。この時、並列カウン

タ２０３の値Ｃは「２」となっているので、並列実行時間テーブル２０２の第１、第２番目のエントリの値Ｔ１、Ｔ２に（ＳＴＭ－ＴＳ）を加算し、その後、タイムスタンプ２０４の値ＴＳにシステムタイマ２０５の値ＳＴＭを設定する。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝３０＋（５０－４０）＝４０、Ｔ２＝Ｔ２＋（ＳＴＭ－ＴＳ）＝１０＋（５０－４０）＝２０、ＴＳ＝ＳＴＭ＝５０となる。次に、並列カウンタ減少処理部１０４が、並列カウンタ２０３の値Ｃを１減少させ、Ｃ＝１とする。

【００７０】

その後、プロセッサ１００－２内の処理部１０６でスレッドＣが生成される。スレッドＣが生成されると、プロセッサ１００－２内のスレッド開始処理部１１０は、スレッドＣを実行待ちキューにつなぐと共に、スレッドＣがプロセスＤ内の最初のスレッドであるか否かを判断する（ステップＳ１）。この場合、スレッドＣは最初のスレッドでないので、ステップＳ２の処理はスキップされる。

【００７１】

また、プロセッサ１００－２内の並列実行時間算出処理部１０７において、並列実行時間の算出処理が実行される（ステップＳ１３）。この時、並列カウンタ２０３の値Ｃが「１」であるので、並列実行時間テーブル２０２の第１番目のエントリの値Ｔ１を更新すると共に、タイムスタンプ２０４の値ＴＳを更新する。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝４０＋（５０－５０）＝４０、ＴＳ＝ＳＴＭ＝５０となる。次に、並列カウンタ増加処理部１０８が、並列カウンタ２０３の値Ｃを１増加させ、Ｃ＝２とする（ステップＳ１４）。その後、割込み終了処理部１０９で割込み終了処理が実行され、スレッドＢは、ユーザプログラムの処理に復帰する。

【００７２】

一方、実行待ちキューにつながれている、新たに生成されたスレッドＣに或るプロセッサ（例えば、プロセッサ１００－３とする）内のプロセッサ割当処理部１１１によりプロセッサ１００－３が割り当てられると、並列実行時間算出処理部１０７が、並列実行時間の算出処理を行う（ステップＳ１３）。この時、並列カウンタ２０３の値Ｃは「２」であるので、並列実行時間テーブル２０２の第１番目、第２番目のエントリの値Ｔ１、Ｔ２が更新されると共に、タイムスタンプ２０４の値ＴＳが更新される。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝４０＋（５０－５０）＝４０、Ｔ２＝Ｔ２＋（ＳＴＭ－ＴＳ）＝２０＋（５０－５０）＝２０、ＴＳ＝ＳＴＭ＝５０となる。次に、並列カウンタ増加処理部１０８が、並列カウンタ２０３の値Ｃを１増加させ、Ｃ＝３とする。

【００７３】

その後、割込み終了処理部１０９が割込み終了処理を行い、スレッドＣはユーザプログラムの実行を開始する。なお、スレッドＢが実行されているプロセッサ１００－２内の並列実行時間算出処理部１０７、並列カウンタ増加処理部１０８よりも先に、スレッドＣが実行されるプロセッサ１００－３内の並列実行時間算出処理部１０７、並列カウンタ増加処理部１０８がステップＳ１３、Ｓ１４の処理を実行したとしても最終的な結果は同じである。

【００７４】

次に、時刻Ｐ６（ＳＴＭ＝６０）において、プロセッサ１００－１で動作しているスレッドＡが、終了するために終了システムコールを発行すると、プロセッサ１００－１内の割込み開始処理部１０２に制御が移り、並列実行時間算出処理部１０３が並列実行時間の算出処理を行う（ステップＳ７）。このとき、並列カウンタ２０３の値Ｃは「３」であるので、並列実行時間テーブル２０２の第１番目、第２番目、第３番目のエントリの値Ｔ１、Ｔ２、Ｔ３が更新されると共に、タイムスタンプ２０４の値ＴＳが更新される。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝４０＋（６０－５０）＝５０、Ｔ２＝Ｔ２＋（ＳＴＭ－ＴＳ）＝２０＋（６０－５０）＝３０、Ｔ３＝Ｔ３＋（ＳＴＭ－ＴＳ）＝０＋（６０－５０）＝１０、ＴＳ＝６０となる。その後、並列カウンタ減少処理部１０４が、並列カウンタ２０３の値Ｃを－１し、Ｃ＝２とする。

【００７５】

その後、プロセッサ返還処理部１１３が、スレッドＡに割り当てていたプロセッサの返還処理を行い、更に、スレッド終了処理部１１４が、スレッドＡの終了処理を行う。このとき、スレッドＡは、プロセスＤ内の最後のスレッドでないので（ステップＳ５がＮｏ）、スレッド共有空間２０１の解放処理は行われない。

【００７６】

次に、時刻Ｐ７（ＳＴＭ＝６５）において、プロセッサ１００－３で動作しているスレッドＣが、終了するために終了システムコールを発行すると、プロセッサ１００－３内の割込み開始処理部１０２に制御が移る。これにより、並列実行時間算出処理部１０３が、並列実行時間の算出処理を行う（ステップＳ７）。このとき、並列カウンタ２０３の値Ｃは、「２」であるので、並列実行時間テーブル２０２の第１番目、第２番目のエントリの値Ｔ１、Ｔ２が更新されると共に、タイムスタンプ２０４の値ＴＳが更新される。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝５０＋（６５－６０）＝５５、Ｔ２＝Ｔ２＋（ＳＴＭ－ＴＳ）＝３０＋（６５－６０）＝３５、ＴＳ＝ＳＴＭ＝６５となる。その後、並列カウンタ減少処理部１０４が並列カウンタ２０３の値Ｃを１減少させ、Ｃ＝１とする（ステップＳ８）。

【００７７】

その後、プロセッサ返還処理部１１３が、スレッドＣに割り当てていたプロセッサの返還処理を行い、更に、スレッド終了処理部１１４が、スレッドＣの終了処理を行う。このとき、スレッドＣは、プロセスＤ内の最後のスレッドでないので（ステップＳ５がＮｏ）、スレッド共有空間２０１の解放処理は行われない。

【００７８】

次に、時刻Ｐ８（ＳＴＭ＝７０）において、プロセッサ１００－２で動作しているスレッドＢが、終了するために終了システムコールを発行すると、プロセッサ１００－２内の割込み開始処理部１０２に制御が移る。これにより、並列実行時間算出処理部１０３が、並列実行時間の算出処理を行う（ステップＳ７）。このとき、並列カウンタ２０３の値Ｃは「１」であるので、並列実行時間テーブル２０２の第１番目の値Ｔ１を更新すると共に、タイムスタンプ２０４の値ＴＳを更新する。すなわち、Ｔ１＝Ｔ１＋（ＳＴＭ－ＴＳ）＝５５＋（７０－６５）＝６０、ＴＳ＝ＳＴＭ＝７０となる。その後、並列カウンタ減少処理部１０４が、並列カウンタ２０３の値Ｃを１減算し、Ｃ＝０とする（ステップＳ８）。

【００７９】

その後、プロセッサ返還処理部１１３がスレッドＢに割り当てていたプロセッサの返還処理を行い、更に、スレッド終了処理部１１４がスレッドＢの終了処理を行う。この時、スレッドＢは、プロセスＤ内で最後のスレッドであるので（ステップＳ５がＹｅｓ）、スレッド終了処理部１１４は、スレッド共有空間２０１を解放する（ステップＳ６）。

【００８０】

以上の処理によりスレッドＡ、Ｂ、Ｃから構成される並列プログラムの並列実行時間が算出でき、任意の時点で発行する並列実行時間取得システムコールにより、その瞬間の並列実行時間（Ｔ１，Ｔ２，Ｔ３）を容易に取得することが可能となる。例えば、時刻Ｐ８でスレッドＢが並列実行時間取得システムコールを発行した場合、スレッドＢは、並列実行時間（Ｔ１＝６０，Ｔ２＝３５，Ｔ１＝１０）を取得することができる。そして、スレッドＢは、上記並列実行時間を取得すると、例えば、取得した並列実行時間を表示部（図示せず）に表示したり、並列実行時間に基づいて２並列度＝３５／６０＝０．５８３３（約５８．３％）、３並列度＝１０／６０＝０．１６６６６（約１６．７％）を算出し、それらを表示部に表示する。

【００８１】

なお、上述した実施例においては、主記憶装置２００上にスレッド共有空間２０１を生成するようにしたが、各プロセッサ１００－１～１００－Ｎによって共有される他の記憶装置上にスレッド共有空間２０１を生成するようにしても構わない。

【００８２】

【発明の効果】

本発明の第１の効果は、マルチプロセッサ計算機上で複数の並列プログラムが動作している場合であっても、各並列プログラムの並列実行時間を取得できるという点である。その理由は、プロセス固有或いはジョブ固有の並列実行時間テーブルを利用して並列実行時間を管理するようにしているからである。

【００８３】

本発明の第２の効果は、並列プログラムの実行時の並列度として、２並列度からＮ並列度までの最大（Ｎ－１）個の並列度が取得でき、並列度の詳細な検証ができる点である。その理由は、複数のスレッド数（１，２，…，Ｎ）毎に、そのスレッド数以上のスレッドに同時にプロセッサが割り当てられた時間を登録する領域を有する並列実行時間テーブルを用いて、並列実行時間を管理しているからである。

【図面の簡単な説明】

【図１】本発明の実施例のブロック図である。

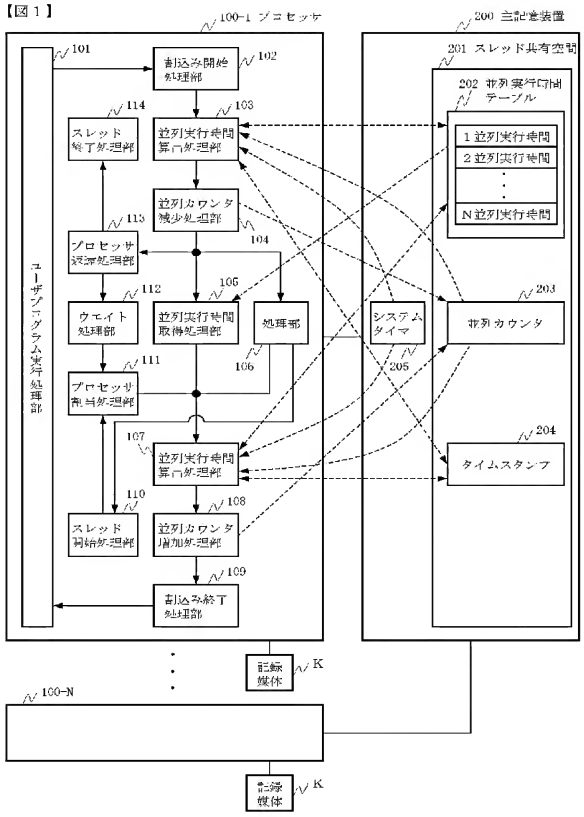
【図２】図１の処理例を示す流れ図である。

【図３】具体例を挙げて実施例の動作を説明するための図である。

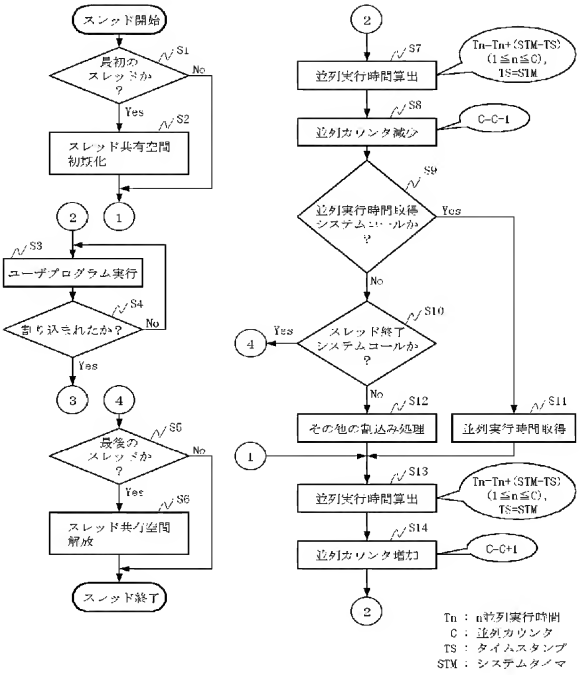
【符号の説明】

１００－１～１００－Ｎ…プロセッサ
１０１…ユーザプログラム実行処理部
１０２…割込み開始処理部
１０３…並列実行時間算出処理部
１０４…並列カウンタ減少処理部
１０５…並列実行時間取得処理部
１０６…処理部
１０７…並列実行時間算出処理部
１０８…並列カウンタ増加処理部
１０９…割込み終了処理部
１１０…スレッド開始処理部
１１１…プロセッサ割当処理部
１１２…ウェイト処理部
１１３…プロセッサ返還処理部
１１４…スレッド終了処理部
２００…主記憶装置
２０１…スレッド共有空間
２０２…並列実行時間テーブル
２０３…並列カウンタ
２０４…タイムスタンプ
２０５…システムタイマ
Ｋ…記録媒体

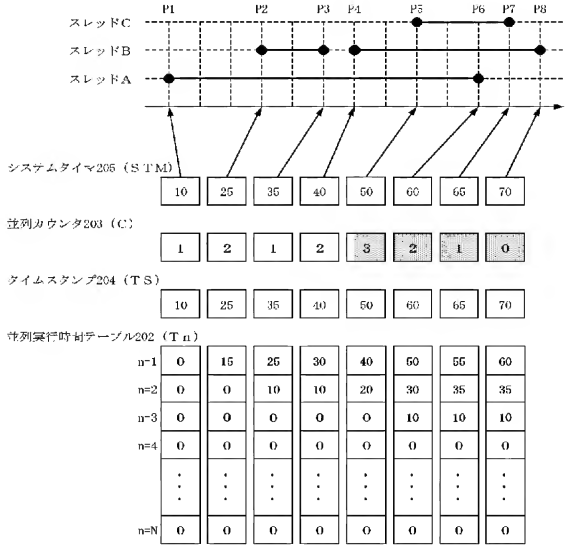
【図１】


【図１】
100-1 プロセッサ
200 主記憶装置
201 スレッド共有空間
101
割込み開始処理部
102
114
スレッド終了処理部
103
並列実行時間算出処理部
202 並列実行時間テーブル
1 並列実行時間
2 並列実行時間
N 並列実行時間
並列カウンタ減少処理部
104
115
プロセッサ選択処理部
112
ウエイト処理部
105
並列実行時間取得処理部
処理部
システムタイマ
203
並列カウンタ
106
205
111
プロセッサ割当処理部
107
並列実行時間算出処理部
204
タイムスタンプ
110
スレッド開始処理部
108
並列カウンタ増加処理部
109
割込み終了処理部
ユーザプログラム実行処理部
記録媒体
K
100-N
記録媒体
K

【図２】


【図２】
スレッド開始
S1
最初のスレッドか？
No
Yes
S2
スレッド共有空間初期化
ユーザプログラム実行
S3
S4
割り込まれたか？
No
Yes
S5
最後のスレッドか？
No
Yes
S6
スレッド共有空間解放
スレッド終了
S7
並列実行時間算出
Tn=Tn+(STM-TS) (1≦n≦C), TS=STM
S8
並列カウンタ減少
C=C-1
S9
並列実行時間取得システムコールか？
Yes
No
S10
スレッド終了システムコールか？
Yes
No
S11
並列実行時間取得
S12
その他の割込み処理
S13
並列実行時間算出
Tn=Tn+(STM-TS) (1≦n≦C), TS=STM
S14
並列カウンタ増加
C=C+1
Tn : n並列実行時間
C : 並列カウンタ
TS : タイムスタンプ
STM : システムタイマ

【図３】


【図３】
P1 P2 P3 P4 P5 P6 P7 P8
スレッドC
スレッドB
スレッドA
システムタイマ205（STM）
10 25 35 40 50 60 65 70
並列カウンタ203（C）
1 2 1 2 3 2 1 0
タイムスタンプ204（TS）
10 25 35 40 50 60 65 70
並列実行時間テーブル202（Tn）
n=1 0 15 25 30 40 50 55 60
n=2 0 0 10 10 20 30 35 35
n=3 0 0 0 0 0 10 10 10
n=4 0 0 0 0 0 0 0 0
n=N 0 0 0 0 0 0 0 0